心智与计算, Vol.1，No.2 (2007), 261-268
文章编号：MC - 2007-024
收稿日期：2007-04-15
出版日期：2007-06-30

基于性能反馈的 PSO 算法速度上限调整策略

蔡星娟[1]，崔志华[2,1]，曾建潮[1]，谭　瑛[1]
（1. 太原科技大学系统仿真与计算机应用研究所，山西 太原 030024；
2. 西安交通大学机械制造系统工程国家重点实验室，陕西 西安 710049）
cai_xing_juan@sohu.com，cuizhihua@gmail.com，zengjianchao@263.net

摘要:速度上限是影响微粒群算法性能的一个主要参数。针对现有调节策略存在参数统一设置、与微粒性能无关等缺点，本文提出一种基于性能反馈的调整策略，使得速度上限能随着个体性能的改变而动态调整，从而更加真实有效的模拟了鸟类觅食的群体行为特征。仿真结果表明该算法能较好地提高微粒群算法的计算效率。

关键词：微粒群算法；速度上限；性能反馈
中图分类号：TP18　　**文献标识码:**A

Velocity Threshold Adjustment Strategy of PSO Based on Performance Feedback

CAI Xing-juan [1], CUI Zhi-hua[1,2], ZENG Jian-chao [2] , TAN Ying [2]
（1. Division of System Simulation and Computer Application, Taiyuan University of Science and Technology, Taiyuan 030024, China； 2. State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China）
cai_xing_juan@sohu.com，cuizhihua@gmail.com，zengjianchao@263.net

Abstract: Velocity threshold is an important parameter in particle swarm optimization. There are some disadvantages for previous proposed strategies such as the same values for the entire swarm, no relationship with particle's performance. Therefore, this paper proposes a adjustment strategy based on performance feedback that the velocity threshold is "particle-dependent". This new method provides a deep insight for the birds seeking behavior model. Simulation results show the new proposed algorithm improves the performance greatly.

Key words: particle swarm optimization; velocity threshold; performance feedback

基金项目：国家自然科学基金（No：60674104）资助。

1 前言

微粒群算法[1-2]（Particle Swarm Optimization, PSO）是一种新型的群智能优化算法，由于该算法概念简单，实现容易，已经成功应用于许多领域[3-6]。其基本微粒群算法的进化公式如下：

$$v_{jk}(t+1) = wv_{jk}(t) + c_1 r_1 (p_{jk} - x_{jk}(t)) + c_2 r_2 (p_{gk} - x_{jk}(t)) \quad (1)$$

$$x_{jk}(t+1) = x_{jk} + v_{jk}(t+1) \quad (2)$$

其中，$v_{jk}(t+1) \le v_{\max}, k = 1,2,\cdots.$，$v_{jk}(t)$ 表示第 t 代微粒 j 速度向量的第 k 维分量，$x_{jk}(t)$ 表示第 t 代微粒 j 位置向量的第 k 维分量，p_{jk} 表示第 t 代微粒 j 历史最优位置的第 k 维分量，g_{gk} 表示种群历史最优位置的第 k 维分量，$v_{\max}$ 为速度上限，r_1、r_2 分别为两个介于 0 与 1 之间的随机数，w 为惯性系数，c_1、c_2 分别为认知系数和社会系数，通常取 2.0。

在(1)式所描述的速度进化方程中，其第一部分为微粒先前的速度，它表明微粒对自身飞行的一种探索，由于仅有上一代的速度信息可以利用，因此具有一定的随机性，可以使微粒跳出局部极值点，具有一定的全局搜索能力。第二部分为“认知”部分，因为它考虑了微粒自身的经验，表示微粒本身的思考。认知部分使得微粒朝着自身最优位置方向靠拢。第三部分为“社会”部分，表示微粒间的社会信息共享。使微粒朝着群体历史最优位置方向靠拢。

从微粒群算法的进化方程可以看出，该算法有四个参数需要调整，即惯性系数 w、认知系数 c_1、社会系数 c_2 及速度上限 $v_{\max}$。目前，绝大部分参数研究主要集中于前三个参数，而对速度上限 $v_{\max}$ 的修改设置则很少有文献涉及，因为它并不直接改变微粒速度的选择，而是通过影响其范围来适当的调整速度。

在微粒群算法的初始阶段，速度上限 $v_{\max}$ 一般为常数，取定义域范围的 $10\% \sim 100\%$[1]。为了研究该参数的选择方式，Y. Shi[7]详细研究了 $v_{\max}$ 取固定常数的作用，结果表明只要增加相应的函数计算次数，速度上限 $v_{\max}$ 可以被忽略，然而此时算法的计算效率较低。为了利用 $v_{\max}$ 提高算法性能，P. Fourie[8]提出了一种速度上限的动态调整策略，该策略设定：若微粒的个体历史最优位置在一定时间内没有发生变化，则减小 $v_{\max}$ 的取值，以提高算法的局部搜索能力。J. F. Schutte[9]针对具体的数值优化问题研究了这种策略。结果表明该策略能有效地提高算法的收敛性能。相比较其它参数（如 c_1, c_2）的改进而言，现有速度上限 $v_{\max}$ 的研究对算法性能的改进幅度并不是很大，因此，一般都将 $v_{\max}$ 作为一个辅助参数，与其它参数一起来对算法性能进行改善。

通过研究，作者发现现有的速度上限 $v_{\max}$ 选择策略具有以下两个缺陷：(1) 速度上限 $v_{\max}$ 统一设置，没有区别对待；(2) $v_{\max}$ 的选择没有与微粒性能相联系，从而在一定程度上减弱了 $v_{\max}$ 对微粒速度的调整能力。因此，本文提出了一种全新的 $v_{\max}$ 的调整策略：基于性能反馈的速度上限调整策略（Velocity Threshold Strategy Based on Performance Feedback, VTPF）。通过微粒性能的反馈，对 $v_{\max}$ 进行相应的调整。实例仿真表明该策略非常有效。

2 VTPF 策略介绍

本文主要讨论如下问题： $\min f(x), x \in D \subseteq R^n$.

2.1 VTPF 设计思想简介

$v_{\max}$ 作为一种隐含的因素，对速度起着一定的限制作用，一般对它进行统一设置，使得所有微粒在进化过程中使用同一个速度上限 $v_{\max}$。从生物学方面分析，当某只鸟发现一个食物源之后，会优先考虑进食，而不是继续寻找一个更大的食物源，此时它就会减小速度，补充体能。而对于那些没有发现较好食物源的鸟而言，为了生存，它会增加速度，以便能较快的发现附近的食物源。这表明对于不同的微粒（个体），由于其发现食物的能力不同，其速度上限 $v_{\max}$ 也应存在差异。因此，针对不同的微粒采用不同的速度上限才能更符合微粒群算法的生物学背景。那么，如何才能体现不同微粒的觅食能力呢？

从微粒群算法数学模型分析，微粒在每一代进化过程中由于当前位置及自身历史最优位置的不同，使得它们表现出不同的性能。因此，一个简单直观的想法就是利用微粒性能来体现不同的觅食能力。

2.2 VTPF 调整策略

为了建立基于性能反馈的微粒上限调整策略，我们需要解决两个问题：(1) 如何定义微粒的觅食能力？(2) 如何利用这种能力调整 $v_{\max}$？

首先解决第一个问题：如何定义微粒的觅食能力？我们知道：微粒当前适应值是微粒比较重要的一个性能。对于当前位置性能较好（适应值较低）的微粒，全局最优值落在它附近的概率较大，此时该位置所含的食物要多于其它微粒所在的位置。因此，该微粒应优先补充食物，从而具有一个较小的速度上限 $v_{\max}$，将其速度限制在一个较小的范围之内，这样，可以让它在自身附近继续寻优。反之，性能稍差（适应值较低）的微粒，全局最优值落在它附近的概率较小，则给定它一个较大的 $v_{\max}$，使得它的速度限制在一个比较大的范围之内，这样该微粒就有能力跳出自身附近进行全局寻优。

因此，当前适应值越高，表明微粒在当前进化代数中发现的食物越多。反之亦然。因此选择微粒的当前适应值可以作为评价微粒性能优劣的主要标准。为了表示方便，设 $X(t)=(x_1(t),x_2(t),...,x_n(t))$ 是第 t 代的群体，$x_j(t)$ 表示微粒 j 在第 t 代的位置向量。$f_{worst}(X(t))=\arg\max\{f(x_j(t))\mid j=1,2,...,n\}$ 为第 t 代群体中最差个体的适应值，而 $f_{best}(X(t))=\arg\min\{f(x_j(t))\mid j=1,2,...,n\}$ 为第 t 代群体中最优的个体的适应值，则可定义微粒 j 在第 t 代的觅食性能指标：

$$score_j(t)=\begin{cases}1, & \text{if}(f_{worst}(X(t))=f_{best}(X(t))),\\ \dfrac{f_{worst}(X(t))-f(x_j(t))}{f_{worst}(X(t))-f_{best}(X(t))}, & otherwise.\end{cases} \tag{3}$$

由式（3）式可以看出第 t 代中，当前适应值越高的微粒，它的觅食性能指标 $score_j(t)$ 越大，而当前适应值越低的微粒，它的觅食性能指标 $score_j(t)$ 越小。觅食性能指标 $score_j(t)$ 主要是由微粒自身的当前适应值来确定的，这样相当于我们给各微粒在第 t 代的适应值进行了排名，使得当前适应值越好的微粒性能越好，反之亦然。

有了以上对微粒觅食性能优劣的评价，下面解决第二个问题。我们给出如下定义：

$$v_{\max,j}(t)=v_{\max,low}(t)+(v_{\max,high}(t)-v_{\max,low}(t))\times(1-score_j(t)) \tag{4}$$

其中，$v_{\max,j}(t)$ 表示微粒 j 在第 t 代的最大速度上限，而 $v_{\max,low}(t)$ 和 $v_{\max,high}(t)$ 分别表示 $v_{\max,j}(t)$ 在第 t 代的上、下限。由公式（4）可以看出，微粒觅食性能越好，也就是 $score_j(t)$ 越大，$v_{\max,j}(t)$ 就越小，反之亦然。由于 $v_{\max,j}(t)$ 的选择依赖于微粒的觅食性能指标，因此，我们称该策略为基于性能反馈的速度上限调整策略。

2.3 基于 VTPF 调整策略的微粒群算法（PSO-VTPF）流程

Step1. 对微粒群中各微粒的初始位置和初始速度进行设定。设各微粒的初始位置均匀分布在 $[-x_{\max},x_{\max}]$，而相应的速度向量则均匀分布在$[-v_{\max},v_{\max}]$中；

Step2. 计算每个微粒的适应值；

Step3. 对于每个微粒 i，将其适应值与个体的历史最优位置 P_i 的适应值进行比较，若更优，则将其作为当前的最好位置；

Step4. 对每个微粒，将其历史最优适应值与群体所经历的最优位置 P_g 的适应值进行比较，若更优，则将其作为当前的群体历史最优位置；

Step5. 按照公式（3）与（4）计算速度上限 $v_{\max,j}$；

Step6. 根据方程（1）计算微粒下一代的速度；

Step7. 根据 Step5 计算的微粒速度上限对各微粒速度进行限定；

Step8. 根据方程（2）计算微粒下一代的位置；

Step9. 如果没有达到结束条件，则返回 Step2；否则，输出最优结果。

3 仿真实例

3.1 测试函数

为了验证本文算法的有效性，我们利用基本微粒群算法（Standard Particle Swarm Optimization, SPSO）及带时间加速常数的微粒群算法[10]（Modified Time-varying accelerator coefficients Particle Swarm Optimization, MPSO_TVAC）与本文提出的新算法 PSO-VTPF 进行了比较。在比较过程中，利用下述四个测试函数进行测试：

(1) Schwefel Problem 2.26

$$f_1(x) = -\sum_{i=1}^{n}\left(x_i \sin(\sqrt{|x_i|})\right), \qquad -500 \le x_i \le 500,$$

$$\min(f_1) = f_1(420.9687,...,420.9687) = 12569.5.$$

(2) Penalized Function

$$f_2(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$$

$$-50 \le x_i \le 50, \qquad \min(f_2) = f_2(1,...,1) = 0.$$

这里

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a, \\ 0, -a \le x_i \le a, \\ k(-x_i - a)^m, x_i < -a. \end{cases}$$

$$y_i = 1 + \frac{1}{4}(x_i + 1).$$

(3) Hartman Family

$$f_3(x) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2], \quad 0 \le x_j \le 1,$$

$$\min(f_3) = f_3(0.114, 0.556, 0.852) = -3.86.$$

表 1 Hartman Family 参数列表

Tab. 1 Hartman Family parameter list

i	a_{ij}, j=1,2,3			c_i	p_{ij}, j=1,2,3		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

(4) Hartman Family

$$f_4(x) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2]. \quad 0 \le x_j \le 1,$$

$$\min(f_4) = f_4(0.201,0.15,0.477,0.275,0.311,0.657) = -3.32.$$

表 2 Hartman Family 参数列表

Tab. 2 Hartman Family parameter list

i	a_{ij}, j=1,…,6						c_i	p_{ij}, j=1,…,6					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

其中， Schwefel Problem 2.26 与 Penalized Function 为多峰具有许多局部极值点的函数，而两个 Hartman Family 函数为多峰则只具有几个局部极值点的函数。

3.2 参数设置

测试函数 f_1 与 f_2 的维数为 30，最大进化代数都设置为 1 000，种群微粒为 100。而 f_3 与 f_4 分别为 3 和 6 维，考虑到 f_3 与 f_4 的维数较小，所以它的最大进化代数设置为 100，种群所含微粒为 20。对于每个测试函数，算法都均运行 30 次。

算法 SPSO、MPSO_TVAC、PSO-VTPF 中惯性系数 w 都随进化代数的增加由 0.9 线性递减到 0.4。

SPSO、PSO-VTPF 的加速度系数 c_1、c_2 都为常数 2.0，而算法 MPSO_TVAC 的加速度系数随着进化代数的增加，c_1 由 2.5 线性递减到 0.5，c_2 由 0.5 线性递增到 2.5。SPSO、MPSO_TVAC 中最大速度常数为定义域的上界 $x_{\max}$，PSO-VTPF 最大速度常数的上界 $v_{\max,high}(t)$ 由定义域的上界 $x_{\max}$ 线性递减到 $0.1*x_{\max}$，下界 $v_{\max,low}(t)$ 为 $0.1*x_{\max}$。

3．3 算法性能分析

我们从在相同进化代数下得到的最优值进行分析。表 1 和图 1 的四个子图分别为各函数在相同进化代数下比较的结果。其中表 1 表示各函数在相同代数下的均值、方差、最优值和最差值。而图 1 的各子图中，横坐标表示进化代数，纵坐标表示平均适应值。通过对表 1 与图 1 分析，有如下结论：

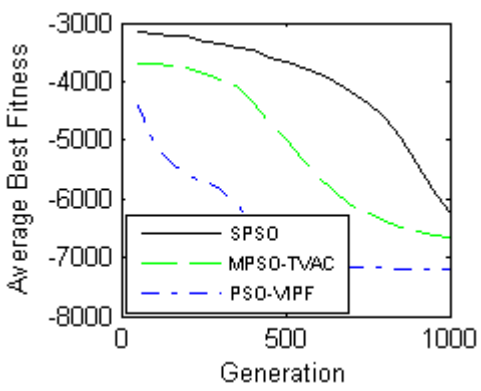
（1）对于多峰且具有多个极值点的函数 f_1 与 f_2：从表 1 可以看出，算法 PSO-VTPF 无论是均值还是方差都远远超过了两个比较算法 SPSO 与 MPSO_TVAC，其中，对于函数 f_1，PSO-VTPF 的均值比 MPSO_TVAC 至少优 8%，而比 SPSO 优 14% ，对于函数 f_2，PSO-VTPF 的均值比 MPSO_TVAC 与 SPSO 至少优 8 个数量级；从图 1 的子图（a)、(b）可以看出，在整个进化过程中，PSO-VTPF 的寻优性能都远远超过了 MPSO_TVAC

与 SPSO。

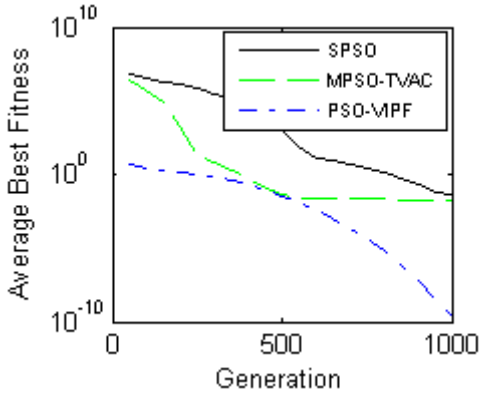
表 3 四个测试函数的静态性能比较结果

Tab.3 The four test functions ' static performance comparison results

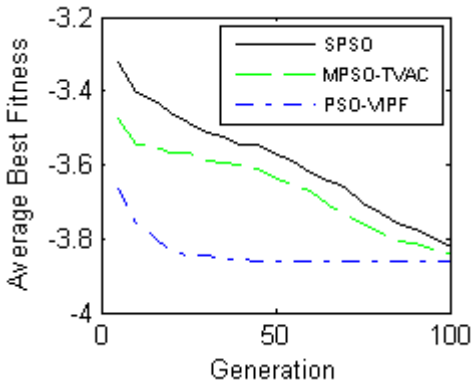
函数	算法	均值	方差	最优值	最次值
F1	SPSO	-6.2474e+003	9.2131e+002	-7.7758e+003	-4.1316e+003
	MPSO-TVAC	-6.6502e+003	6.0927e+002	-8.6029e+003	-5.6903e+003
	PSO-VTPF	-7.1937e+003	7.3959e+002	-9.4109e+003	-5.7786e+003
F2	SPSO	4.3043e-002	6.6204e-002	5.7290e-007	2.0736e-001
	MPSO-TVAC	1.72786e-002	3.9295e-002	3.4548e-024	1.0366e-001
	PSO-VTPF	1.8918e-010	1.9135e-010	2.4305e-012	6.9232e-010
F3	SPSO	-3.8248e+000	3.8934e-002	-3.8626e+000	-3.7340e+000
	MPSO-TVAC	-3.8404e+000	3.7203e-002	-3.8626e+000	-3.6763e+000
	PSO-VTPF	-3.8627e+000	5.7825e-009	-3.8627e+000	-3.8627e+000
F4	SPSO	-2.6217e+000	2.6712e-001	-3.1257e+000	-2.0645e+000
	MPSO-TVAC	-2.9926e+000	2.3679e-001	-3.3176e+000	-2.3954e+000
	PSO-VTPF	-3.2594e+000	6.3845e-002	-3.3219e+000	-3.1752e+000



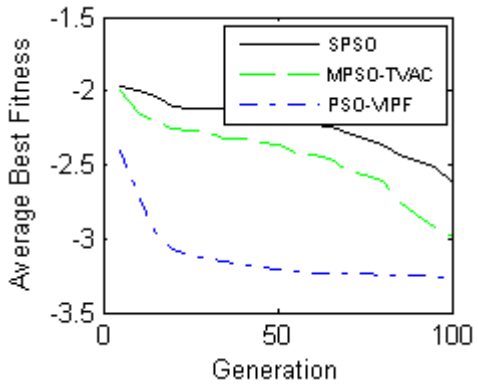
（a） f_1 的比较结果



（b） f_2 的比较结果



（c） f_3 的比较结果



（d） f_4 的比较结果

图 1 四个测试函数的动态性能比较图

Fig.1 The four test functions ' dynamic performance comparison charts

（2）对于多峰且具有几个极值点的函数 f_3 与 f_4：表 1 显示 PSO-VTPF 具有非常高的计算性能，对于函数 f_3，PSO-VTPF 的均值几乎达到了理论值-3.86，而对于 f_4，PSO-VTPF 的均值也远远超过了 MPSO_TVAC 与 SPSO，与理论值-3.32 也比较接近。从图 1 的子图（c）、（d）中也可以看出，PSO-VTPF 的寻优性能无论是在进化初期还是后期，它的均值都与那远超过了 MPSO_TVAC 与 SPSO。

通过以上分析，算法 PSO-VTPF 非常适合多峰函数，尤其适合多峰且具有几个极值点的函数。

4 结论

现有诸多微粒群算法文献中，速度上限 $v_{\max}$ 的设置与微粒的性能无关。通过对基本微粒群算法生物学模型以及数学模型分析，本文提出了一种基于性能反馈的速度上限的调节策略（VTPF），并将其应用于微粒群算法（PSO-VTPF）。仿真结果表明，PSO-VTPF 的计算性能在一定程度上远远超过了基本微粒群算法 SPSO 和 MPSO_TVAC。其后续工作包括如何进一步提高算法计算速度的同时，保持种群多样性，以提高算法的全局搜索能力，减少过早收敛现象的发生。

参考文献：

[1]Kennedy J, Eberhart R. Particle swarm optimization[C]// Proceedings of IEEE International Conference on Neural Networks , Perth,1995, 1942-1948.

[2]Eberhart R, Kennedy J. A new optimizer using particle swarm theory[C]//Proceedings of 6th International Symposium on Micro Machine and Human Science, Nagoya, 1995,39-43.

[3]Salerno J. Using the particle swarm optimization technique to train a recurrent neural model[C]// Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence, 1997, 45-49.

[4]Zhang W, Liu Y T. Adaptive particle swarm optimization for reactive power and voltage control in power systems[C]// Lecture Note in Computer Science, 2005, 3612:449-452.

[5]Sousa T, Silua A, Neves A. A particle swarm data miner[C]// Proceedings of 11th Protuguese Conference on Artificial Intelligence, 2003, 43-53.

[6]Li Q Y, Shi Z P, Shi J, et al. Swarm intelligence clustering algorithm based on attractor[C]//Lecture Note in Computer Science, 2005,3612:496-504.

[7]Shi Y, Eberhart R. Parameter selection in particle swarm optimization[C]// Lecture Notes in Computer Science, 1998, 1447: 591-600.

[8]Fourie P, Groenwold A. The particle swarm optimization algorithm in size and shape optimization[J]. Struc Multidisc Opim, 2001, 23: 259-267.

[9]Schutte J. Particle swarms in sizing and global optimization[D]. University of Pretoria, Department of Mechanical Engineering, 2002.

[10]Ratnaweera A., Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm opitmizer with time-varying acceleration coefficients[J]. IEEE Transactions on Evolutionary Computation, 2004,8(3):240-255.

作者简介：

蔡星娟(1980-)，女，硕士研究生，主要研究方向为智能计算。